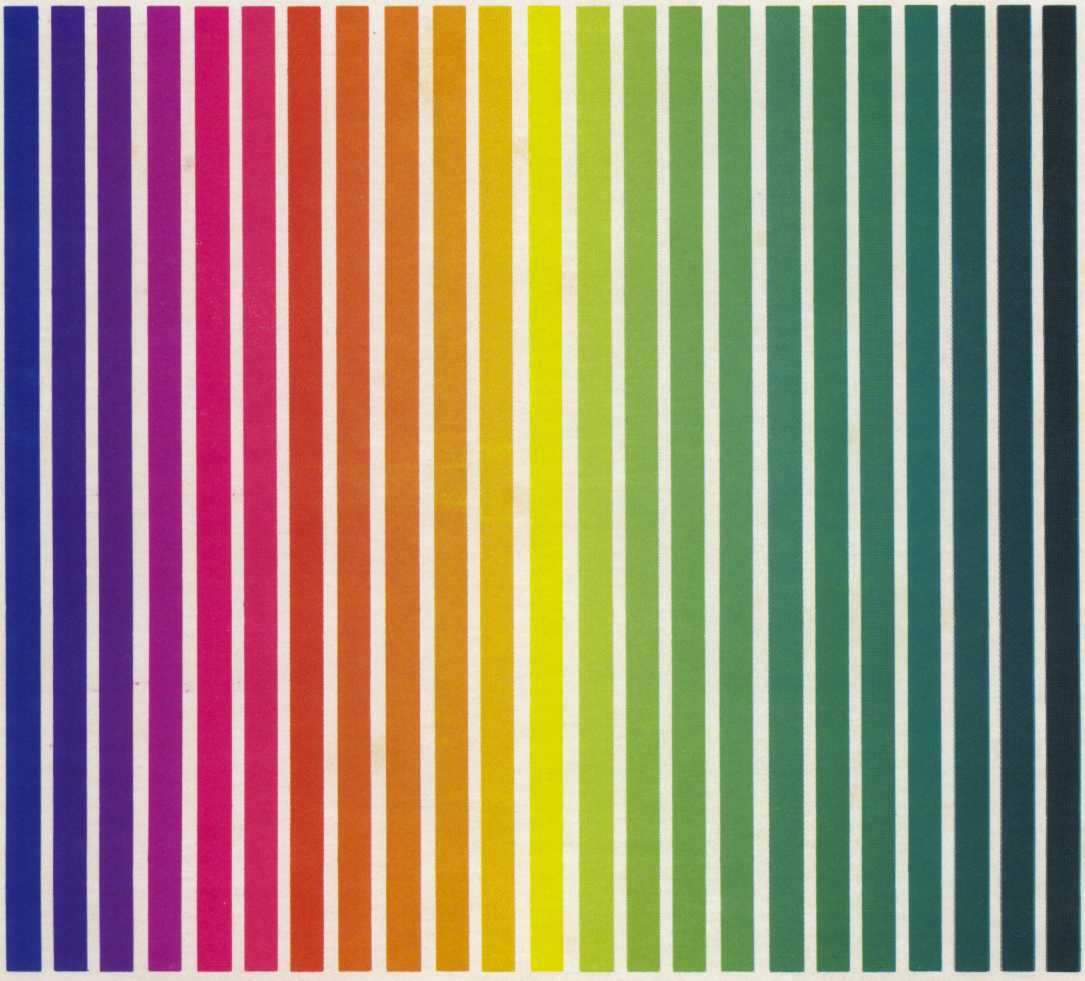
APX ATARI® PROGRAM EXCHANGE



The Soft Warehouse

ALGICALC™

Perform symbolic algebra and some calculus (ages 14 and up)

Cassette: 24K (APX-10126) Diskette: 32K (APX-20126)

User-Written Software for ATARI Home Computers

The Soft Warehouse

ALGICALC™

Perform symbolic algebra and some calculus (ages 14 and up)

Cassette: 24K (APX-10126) Diskette: 32K (APX-20126)

Trademark Notice

ALGICALC is a trademark of The Soft Warehouse.

Disclaimer

The Soft Warehouse makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Further, The Soft Warehouse reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of The Soft Warehouse to notify any person or organization of such revision or changes.

ALGICALC

by

The Soft Warehouse

Distributed By

The ATARI Program Exchange
P.O. Box 3705
Santa Clara, CA 95055

To request an APX Product Catalog, write to the address above, or call toll-free:

800/538-1862 (outside California)
800/672-1850 (within California)

Or call our Sales number, 408/727-5603

Trademarks of Atari

The following are trademarks of Atari, Inc.

ATARI®
ATARI 400™ Home Computer
ATARI 800™ Home Computer
ATARI 410™ Program Recorder
ATARI 810™ Disk Drive
ATARI 820™ 40-Column Printer
ATARI 822™ Thermal Printer
ATARI 825™ 80-Column Printer
ATARI 830™ Acoustic Modem
ATARI 850™ Interface Module

Printed in U.S.A.

CONTENTS

INTRODUCTION

OVERVIEW

ALGICALC is a tool for quickly performing operations in symbolic algebra and calculus. Students can use the program to check the solutions to homework problems. Professionals will find ALGICALC useful for such applications as repetitive symbolic operations. Examples include differentiating expressions and expanding or factoring them over a common denominator.

With ALGICALC, you can perform operations on expressions that can be expressed as a ratio of two polynomials in the variable x. This feature is an advantage over other programming languages requiring numeric values to be assigned to all variables in expressions before evaluating the expression.

ALGICALC is easy to use and is best learned by entering problems and studying the results. If you need help, you can request any of the program's brief built-in instructions at any time. Refer to this manual when you need more information about ALGICALC's conventions and notation than appears in the lessons. You use ALGICALC by typing in an expression, using ALGICALC's special symbols to indicate the kind of operation desired--expansion, factoring, or differentiation. After the results display, you can enter another expression. In addition, you can assign the results to variables for use in later expressions, letting you perform a sequence of related operations.

To use this manual, you should be familiar with first-year high school algebra. In addition, the section on using ALGICALC for performing differentiation assumes you're familiar with first-year calculus.

REQUIRED ACCESSORIES

ATARI BASIC Language Cartridge

Cassette version

24K RAM
ATARI 410 Program Recorder

Diskette version

32K RAM
ATARI 810 Disk Drive

CONTACTING THE AUTHOR

Users wishing to contact the author about ALGICALC may write to The Soft Warehouse at:

P. O. Box 11174
Honolulu, Hawaii 96828

GETTING STARTED

LOADING ALGICALC INTO COMPUTER MEMORY

1. Insert the ATARI BASIC Language Cartridge in the cartridge slot of your computer.

2. If you have the cassette version of ALGICALC:

   a. Turn on your TV set.

   b. Connect your program recorder to the computer and to a wall outlet.

   c. Slide the ALGICALC cassette into the program recorder's cassette holder and press REWIND on the recorder until the tape rewinds completely. Then press PLAY.

   d. Turn on your computer. When the READY prompt appears, type CLOAD, and press RETURN twice. When READY appears again, type RUN.

If you have the diskette version of ALGICALC:

   a. Turn on your disk drive.

   b. When the BUSY light goes out, open the disk drive door and insert the ALGICALC diskette with the label in the lower right-hand corner nearest to you. Close the door.

   c. Turn on your computer and TV set. ALGICALC will load and run automatically.

FIRST DISPLAY SCREEN

The first program display contains the title and copyright information, together with a prompt for requesting help:

```
 ---------------------------------------
|ALGICALC(tm)                           |
|Rational Algebra and Calculus Program  |
|Atari BASIC version                    |
|Copyright (c) 1981 The Soft Warehouse  |
|Box 11174, Honolulu, Hawaii 96828      |
|                                       |
|Enter a problem, or                    |
|enter another question mark for help   |
|                                       |
|?_                                     |
 ---------------------------------------
```

After the title screen displays, you may either start entering problems immediately or select the HELP menu to learn use of ALGICALC's features.

USING ALGICALC

ENTERING PROBLEMS

It's easy to use ALGICALC for solving problems. You type in the numbers, variables, and operators, using the conventions and notation described in the HELP screens and in this manual. If you make a typing mistake, use the DELETE/BACK S key to erase the error and type in the correction. Once you've entered the problem accurately, you press the RETURN key to signal the program to process your problem. ALGICALC beeps when it completes the operation and displays the results. This feature lets you turn your attention elsewhere when ALGICALC is doing time-consuming operations, since it notifies you audibly when you can enter another poblem.

INTERRUPTING COMPUTATIONS

To interrupt a computation, press the BREAK key. This action stops not only the current computation but also the program's functioning. At this point, type GOTO 1920. The program will ask you to enter another problem.

THE KINDS OF OPERATIONS ALGICALC CAN PERFORM

ALGICALC can perform symbolic algebra and calculus operations on expressions that can be expressed as a ratio of two polynomials in the variable x. For example, ALGICALC can reduce this expression:

$$1 + \frac{1}{x - 1} + \frac{x}{x^2 - 1} + \frac{1}{x^2 - 1}$$

into the equivalent expression

$$\frac{x + 1}{x - 1}$$

This reduction involves not only placing expressions over a common denominator, but also canceling the greatest polynomial divisor, x+1, from intermediate subexpressions such as

$$\frac{x^2 + 2x + 1}{x^2 - 1}$$

As another example, ALGICALC can factor

$$\frac{3x^5 + 3x^4 - 258x^3 - 570x^2 + 5775x + 18375}{11x^5 + 110x^4 + 440x^3 + 880x^2 + 880x + 352}$$

into

$$\frac{3\ (x-7)^2\ (x+5)^3}{11(x+2)^5}$$

ALGICALC can also determine that the partial derivative of the above expression with respect to x is

$$\frac{27\ (x+17)\ (x-7)\ (x+5)^2}{11\ (x+2)^6}$$

THE HELP TOPICS

INTRODUCTION

To familiarize yourself with ALGICALC's features before working on your own problems, you can request any of seven instruction screens. Each HELP topic explains a convention ALGICALC follows or a notation it uses. The HELP screens also display one or two problems you can type in for practice. Each topic presumes you are familiar with the material covered in the preceding topics, so it's a good idea work through the HELP screens in order. After you read an explanation, try the practice problems or make up problems of your own. You can also return to the HELP menu at any time by typing a question mark and pressing the RETURN key. Or, you can go directly to another topic by typing the lesson number and a question mark and then pressing the RETURN key.

The seven topics displayed on the HELP menu are as follows:

```
The help topics are

1: Expressions          5: Compound lines
2: Factored display     6: Substitution
3: Assignments          7: Derivative
4: Non-display

Enter a topic number followed by a
question mark.

Study topics in order first time!

See ALGICALC manual for more detail.

?_
```

To select a topic, type its corresponding number and a question mark, and then press the RETURN key. For example, to select Expressions, type:

```
1? [RETURN]
```

The display screen looks like this:

```
After each question mark prompt enter
an expression using digits, decimal
points, parentheses and the letter x
together with operators +, -, /, *
and ^. The latter means raising to a
power.  *, meaning multiply, can be
omitted.

An equivalent expression is displayed
expanded and reduced over a common
denominator, provided the input and
each of its subexpressions can be
represented as a ratio of two
polynomials in x.  For example, type

1 + (x^2 - 1)/(x^2 + 2 + 1)
1.5*10^2/(x+1)^6
4/6

After trying several such examples,
enter a question mark to display the
help topic menu.

?_
```

To try the practice problems, simply type one in and press the RETURN key. ALGICALC will display the results and you can verify your understanding of the lesson.

FORMAT OF DISPLAYED RESULTS

If you type in the first practice problem in lesson one, the result displays as:

```
2x
-------------------------------------
x + 1
```

For each expression you type in, ALGICALC displays the results as the corresponding expanded expression reduced over a common denominator. Notice that the numerator and denominator are separated by a horizontal bar occupying the full width of the screen even if neither value takes up the full width. However, ALGICALC doesn't display the horizontal bar and the denominator when the denominator is 1. Also, ALGICALC displays any result that is a numeric fraction in both fractional and floating-point form. For example, the problem:

```
? 4/6
```

displays as:

```
2
---------------- = 0.6666666666
3
```

The numerator or denominator will continue on the next line when either takes up more than one line, for example:

```
150
-----------------------------------------
x^6 + 6x^5 + 15x^4 + 20x^3 + 15x^2 +
6x+1
```

The following sections contain more information about the lessons, along with explanations of ALGICALC's other features.

EXPRESSIONS

Variables

You can use any letter in the alphabet except X to represent a variable to which you assign a value. Thus, you can use as many as 25 such variables. ALGICALC interprets uppercase and lowercase letters to represent the same thing, so you may type in your variables either way. You may use only the variable X as an unassigned variable. (See the section on assignment for more information.)

Constants

Constants consist of adjacent digits, with an optional decimal point. Examples of constants are 82, 82., 8.2, .82, and 0.82 .

Expressions

Expressions are formed largely following standard algebraic notation. However, ALGICALC does use slight modifications so you can enter data sequentially on one line. The conventions are as follows.

Arithmetic operators

Arithmetic operators appear between two subexpressions. Use + for indicating addition, - for subtraction, / for division, and * for multiplication. Use ^ to raise an expression to a power (called "exponentiation"). For example, X^2 represents X squared (X^2).

Multiplication operator

In contrast to most programming languages, ALGICALC multiplies adjacent subexpressions, whether or not you use an asterisk (*). For example, 5.2*37*X*4 and 5.2 37X4 both represent the product of four quantities, 5.2, 37, X, and 4.

You may use any number of blanks anywhere within or between subexpressions, except that you can't insert blanks within a constant. That is, ALGICALC interprets 32 differently than it interprets 3 2, but it interprets 3X the same as 3 X . However, you don't have to use blanks except between adjacent constants that the program would otherwise interpret as a single constant. For example, to multiply 5 by 6, type 5 6 or 5*6 rather than 56 .

Scientific notation

You can use multiplication together with exponentiation to enter a number in scientific notation. For example, you can enter 8.7*10^11 as 8.7 10^11. Since most integers this large aren't exactly representable in ATARI BASIC, the resulting display is 8.7E+11. In this context, ATARI BASIC uses the letter "E" to represent "*10^" where numerous zeros would otherwise be required to denote the number. However, you can't use this E-notation to enter problems in ALGICALC because it conflicts with the convention of implied multiplication and with using the letter "E" as a variable. ALGICALC output can contain only X as a variable, so E-notation shouldn't cause confusion in ALGICALC output. Nevertheless, you may prefer to avoid using the letter "E" as a variable to avoid confusion.

Signed numbers

A + or - appearing to the left of a variable or constant represents the sign of the number. For example, you enter a negative 3 as -3 . Although a + has no arithmetic effect, you can also enter it if you wish. For example, typing in 8/+--+-4, results in -2 .

Order of operations

When no parentheses indicate otherwise, ALGICALC processes operators in this order:

```
first:   ^
         + and - (when used as signs preceding a number)
         * and /
last:    + and - (when used between subexpressions)
```

For example, ALGICALC evaluates the expression -2^2 as -(2^2), or -4, rather than as (-2)^2, or 4.

Multiple exponentiation

When no parentheses indicate otherwise, ALGICALC computes successive exponentiations right to left. For example, 2^3^2 means 2^(3^2), or 2^9, which is 512, rather than (2^3)^2, or 8^2, which is 64.

Multiple multiplication and division

When no parentheses indicate otherwise, ALGICALC computes successive multiplications and divisions left to right. For example, 6/3*2 means (6/3)*2, or 4, rather than 6/(3*2), or 1 .

Multiple addition and subtraction

When no parentheses indicate otherwise, ALGICALC performs successive additions and subtractions left to right. For example, 6-3+2 means (6-3)+2, or 5, rather than 6+(3-2), or 7 .

Indicating division

In ALGICALC you use the operator / to indicate division in a one-line format rather than building a fraction on more than one line. Therefore, you must use parentheses to get the desired effect when numerators or denominators have more than one term. For example, you enter

```
X + 3
-----
X + 7
```

as (X+3)/(X+7), but not as X+3/X+7, which ALGICALC would interpret as

$$X + \frac{3}{X} + 7$$

or as (X+3)/X+7, which ALGICALC would interpret as

$$\frac{X+3}{X} + 7$$

or as X+3/(X+7), which ALGICALC would interpret as

$$X + \frac{3}{X+7}$$

Similarly, you must use parentheses for denominators of more than one factor unless you use repeated division to get the desired effect. For example, you can enter the fraction

$$\frac{3\ X}{6\ X} \quad \text{or } 1/2$$

as 3X/(6X) or as 3X/6/X (repeated division), but not as 3X/6X, which ALGICALC would interpret as

$$3 * \frac{X}{6} * X \quad \text{or } x^2/2$$

Integer exponents

You may use only whole numbers as exponents in ALGICALC. The program is essentially a rational algebra system rather than a fractional power or exponential system.

IMPROPERLY FORMED EXPRESSIONS

Whenever ALGICALC encounters an improperly formed expression, it displays a ^ underneath the character at which it could no longer continue processing the expression left to right. Then ALGICALC displays a brief error message indicating what kind of data it expected at that point. The program sometimes also reminds you of the availability of the appropriate lessons. The message and location of the ^ often make the cause of the error obvious and you may not need to review conventions and notation. An example of a faulty entry and ALGICALC's response is as follows:

```
? 3 X + 5)^2
          ^

ERROR, expected RETURN, semicolon,
ampersand, operator, digit, decimal
point, variable or (

Enter a problem, or
enter another question mark for help.
```

It may be that the right parenthesis was unintentional or that an earlier left parenthesis was accidentally omitted. Thus, the cause of the error could be almost anywhere left of the indicated stopping point rather than at it. ALGICALC can't read your mind to determine what you really meant! You'll need to re-enter the expression, corrected as appropriate.

UNSTORABLE EXPRESSIONS

ALGICALC stores expressions as a ratio of two polynomials in X. Whenever it encounters an expression it can't represent in this form, ALGICALC displays an appropriate error message and a ^ underneath the character at which it stopped processing the expression left to right. Here are two examples of unstorable expressions and ALGICALC's response:

```
? 2^X
    ^

ERROR, expected numeric power because
 base <> 1

Enter a problem, or
enter another question mark for help.
```

and

```
? (X+1)^0.5
          ^

ERROR, expected integer powered result

Enter another question mark for help.
```

Each subexpression of your problem must be representable as a ratio of two polynomials in X, not just the final result. For example, although the product

$$(X+1)^{0.5}(X+1)^{1.5}$$

would be representable after collecting similar factors, ALGICALC stops processing the expression and then displays an error message after discovering the first fractional power.

Recasting problems to satisfy ALGICALC

A simple manual change of variable can often recast a problem into a form suitable for ALGICALC. For example, if you replace c^x with X, you transform the expression

$$1 + \frac{c^x (c^x + 1)^2 + c^{4x} + 1}{c^{4x} - 1}$$

into

$$1 + \frac{x(x + 1)^2 + x^4 + 1}{x^4 - 1}$$

which ALGICALC simplifies to

$$\frac{2X^2 + X}{X^2 - 1}$$

Then, you can manually resubstitute c^x for X to obtain the final result:

$$\frac{2\,c^{2x} + c^x}{c^{2x} - 1}$$

FACTORED DISPLAY

You can type in an expression and have ALGICALC display the corresponding reduced expression factored into linear factors over a common demoninator. To use this feature, end the expression with an ampersand (&). For example, if you enter:

```
? (X^4) - 1) / (X^3 + 2X^2 + X + 2) &
```

ALGICALC will display the reduced expression as:

```
(X + 1) (X - 1)
---------------------------------------
(X + 2)
```

Delays in obtaining results

ALGICALC always fully expands expressions reduced over a common denominator while the program initially scans the expressions. ALGICALC also saves results in fully expanded form. The reduction involves a more efficient and accurate method than factoring followed by cancellation of identical factors. Thus, ALGICALC factors only after a full expansion and only for purposes of displaying the solution in factored form. The program signals completion of the expansion-reduction process by scrolling the display up one line. Factoring often takes a significant additional amount of time. For this reason, when ALGICALC can't determine a factor within a few minutes, the program displays the question:

NO CONVERGENCE YET. Continue until BREAK (Y/N)?

Type N in response to the prompt to abort the factorization attempt and display the product of all remaining factors as a single factor. Type Y to indicate you want ALGICALC to continue seeking a factor. Typing N will continue the program.

Imaginary numbers

If a polynomial in X has real coefficients, then it's always theoretically possible to factor it into a product of linear and quadratic polynomials having real coefficients. Moreover, the sums of the degrees of the factors equals the degree of their product. However, applying the quadratic formula to some of these quadratic factors may involve taking a square root of a negative number, thus yielding a so-called imaginary number. This imaginary number is usually expressed as a multiple of the square root of -1, which is abbreviated by the letter "i". ALGICALC also follows this convention. For example, if you type in the expression:

```
? X^3 + 3X^2 + 7X + 5 &
```

ALGICALC displays the result as:

```
(x + 1) (x + 1 + 2i) (x + 1 - 2i)
```

The example shows that resulting factors including imaginary numbers always occur in pairs that differ only in the sign of the coefficient of i (that is, +2i and -2i). If you're unfamiliar with imaginary numbers and would rather know the corresponding quadratic factor for each such pair, just evaluate the specific numeric values of 2a and a^2 + b^2 for use in the formula

```
(X + a + bi) (X + a - bi) --> [X^2 + 2aX + (a^2 + b^2)]
```

Thus, for a result such as (x + 1) (x + 1 + 2i) (x + 1 - 2i), we can manually combine the last two factors into (x^2 + 2x + 5) if desired.

ARITHMETIC LIMITATIONS

To compute the coefficients and exponents of simplified results, ALGICALC uses ATARI BASIC's floating-point arithmetic. This arithmetic limits the precision of numbers. Thus, roundoff errors can accumulate with every step of the expansion process. During expansion, reduction, and even part of the factoring process, ALGICALC attempts to avoid or at least postpone the onset of roundoff errors by converting all input decimal fractions to ratios of integers reduced to lowest terms. It then works entirely with reduced integer coefficients. However, most integers whose magnitude exceeds 9,999,999,999 cannot be represented exactly in ATARI BASIC. Hence, if any coefficients exceed this magnitude, a result may be, and probably is, inexact. Unless the coefficients and degrees of the input polynomials are quite modest, it's easy for coefficients of intermediate or final results to exceed this magnitude. For example, ALGICALC displays the result of this problem:

```
(123456x + 789012)^2
```

as

```
1.52413839E+10x^2 + 1.948165309E+11x +
  6.225399361E+11
```

The relative roundoff errors in the resulting coefficients are often quite small, such as in this example. However, when a coefficient results from the subtraction of two almost-equal inexact coefficients of much larger magnitudes, then the resulting relative error in a coefficient can be quite large.

The final stage of factorization uses an iterative method. Though usually successful, this method isn't guaranteed to converge to an answer. Even when the method does converge, the accuracy may be poor relative to the accuracy of the coefficients in the expanded and reduced result. For example, the method may do poorly when factors are nearly identical. Thus, you may occasionally have to abandon an attempted factorization or reject the displayed factors as being insufficiently accurate. As a consolation, be grateful the accuracy and likelihood of convergence are probably better than for most methods.

Limitations on magnitudes of numbers

Floating-point arithmetic also restricts the largest and smallest nonzero magnitudes of numbers. For ATARI BAISC, these magnitudes are about 10^98 and 10^-98.

ATARI BASIC automatically produces 0 in place of an unrepresentably small-magnitude, nonzero result. This "underflow" treatment may or may not seriously not affect a result. A factor or term attributable only to roundoff error might be destroyed by luck, but some legitimate terms might be destroyed as well.

When a result has an unrepresentably large magnitude, ALGICALC displays the error message

```
ERROR, expected easier problem.
```

You can't do much about overflow except to work with less ambitious problems. However, sometimes you can reformulate a problem by changing a variable so as to involve lower degree and coefficient magnitudes closer to 1. Both actions reduce the likelihood of overflow.

ASSIGNMENTS

Be sure you have a good understanding of the concepts in the previous sections and have done sufficient practice problems before tackling assignments.

Naming and saving a result for use in later expressions is often desirable for the following reasons:

1. Fitting an entire expression in a single, one-line entry may be othewise impossible.

2. Entering correctly a sequence of short formulas may be easier than entering an equivalent lengthy one.

3. Naming and saving a subexpression that occurs several places within an expresson or that is to be used in several different expressions avoids repetitious entry.

ALGICALC uses the following form for assignments:

```
letter = expresson
```

where the letter can be any letter other than X.

The effect of an assignment is to evaluate the expression and then store the resulting value as that of the variable to the left of the equal sign. This value replaces any former value of that variable. ALGICALC displays the resulting expanded value the same as if only the expression were entered.

The variable X and all other variables

The variable X always has its name as its value. In contrast, the other 25 variables (A-W and Y-Z) have no initial values, so using them in an expression before you've assigned them values results in an error. (Note that this is different than for the companion APX program, POLYCALC). As described in the discussion titled "Expressions", ALGICALC expands and reduces each expression over a common denominator immediately after you enter it. Whenever ALGICALC meets a variable in your entered expression during this expansion process, it uses the variable's current value. For ALGICALC, expansion and reduction combined with replacement of variables by their current values is called "evaluation".

Examples

Suppose we enter the assignment

```
? Y = X + 3
```

causing ALGICALC to display the new value of Y:

```
X + 3
```

Next, suppose we enter the assignment

```
? Z = Y^2
```

causing ALGICALC to display the new value of Z as:

```
X^2 + 6 X + 9
```

Now, here's a subtle point vital for understanding assignments. Suppose next we enter the assignment

```
? Y = X + 5
```

causing ALGICALC to display the new stored value of Y as:

```
X + 5
```

If we then enter the expression

```
? Z
```

the resulting display is <u>still</u>

```
X^2 + 6 X + 9
```

despite the changed value of Y and the fact that the assigned value of Z originally depended on Y. This phenomenon occurs because after contributing the value X+3 to the expression Y^2, all traces of Y have disappeared from the result X^2+6X+9, which was assigned to Z. For efficiency, ALGICALC keeps no record of the fact that Z received its value from the evaluaton of Y^2. Therefore, later changes to Y don't affect Z.

> Despite the use of an equal sign, you should regard a sequence of assignments as a sequence of <u>updates</u> to memory locations rather than as a set of simultaneous or independent equations.

This treatment is why only a variable is allowed on the left side of an assignment. Using a character other than an equal sign for assignment would be less confusing, but ALGICALC follows BASIC's and FORTRAN's use of the equal sign for assignment for the sake of familiarity.

You can assign a variable an expression containing the same variable. Study this sequence, for example:

```
? Y = X + 1

X + 1

? Y = Y + 1

X + 2

? Y^2

X^2 + 4X + 4
```

Although the result is as expected based on the discussion above, you may still be surprised at first. Thus, you might prefer to avoid the practice as much as possible by introducing additional variable names to achieve the intended effects.

However, assignments containing the same variable on both sides are useful for incrementing the numerical value of a variable as in this example:

```
? K = 1

1
```

```
 . . .
? K = K + 1
2
 . . .
? K = K + 1
3
```

and so on, where ". . ." represents computations involving K and perhaps other variables.

If you don't intend to use an algebraic or numeric result in later expressions, you needn't waste memory space by saving the result as the value of a variable. In such cases, entering the desired expression is easier and more sensible. Moveover, you can reduce space used by an assigned value of no more interest by reassigning the variable a simple numeric value such as 0.

SUPPRESSED DISPLAY AND COMPOUND ENTRIES

In sequences such as those of the previous subsection, intermediate assignments occur for which the resulting display is of no interest. In such cases, you may want to suppress the displays to avoid their distraction and to delay expressions scrolling off the top of the display. You can omit displaying any expression or assignment by ending it with a semicolon (;).

You can also use the semicolon to enter more than one expression and assignment per line by separating them with semicolons. For example, the entry

```
? W=X-1; Y=W+3; Z=Y^2
```

results in the single display

```
X^2 + 4 X + 4
```

Try practicing using display suppression and compound entry before going on to the next section.

SUBSTITUTIONS

After deriving an expression containing the unassigned variable X, substituting numerical values for X in the expression is often desirable. For example, suppose that after we type the assignment:

```
? R = ((3+X)^3 - (3-X)^3) / (5 + 1/(x^2+3))
```

and ALGICALC displays the result

```
2X^5 + 60X^3 + 162X
-------------------
5X^2 + 16
```

we wish to know the values of R corresponding to X=1, X=2, ..., X=9. The operator named "@" lets us do this conveniently as follows:

```
? R @ 1

32
---------- = 10.66666666
3

? R @ 2

217
---------- = 24.11111111
9

 ...

? R @ 9

163296
---------- = 387.876484
421
```

In general, using the @ operator has the form

$$\text{expression}_1 \ @ \ \text{expression}_2$$

where expression_2 must reduce to a numeric value. The effect is to produce the number obtained by substituting the value of expression_2 for x in expression_1.

The operator @ has the same precedence and left-to-right order as addition and subtraction. For example, the result of the problem:

```
? X + 5 @ 2 + X
```

means 2 + 5 + x, resulting in

```
X + 7
```

(Note that @ is used somewhat differently than for the companion APX program, POLYCALC.)

USING THE CALCULUS FEATURE

INTRODUCTION

The following discussion assumes you're familiar with first-year calculus. You can ignore this section if you don't intend to use the calculus feature.

DERIVATIVES

The operator % following an expression or subexpression denotes the derivative of the expression or subexpression with respect to x. (Note that % is used slightly differently than for the companion APX program, POLYCALC.) To help remember it, note that the percentage symbol suggests a ratio of infinitesimals. The precedence and left-to-right direction of evaluation of this operator are the same as for addition. Try this example:

```
? 1/(x^2+3) %

-2 X
----------------------------------------
X^4 + 6 X^2 + 9
```

You can specify higher order derivatives by repeating the operator. For example, to add 5 onto the second order derivative of

$$x^3 + x^2$$

with respect to x, you could enter either

```
? X^3 + x^2 % % + 5
```

or

```
? 5 + (x^3 + x^2 % %)
```

Parentheses are necessary in the second case because addition and differentiation have the same precedence and are done left to right.

HOW TO LEARN MORE ABOUT COMPUTER ALGEBRA

We expect that many who use ALGICALC or another computer algebra system will want to learn more about this fascinating subject or will want to try other systems that have complementary or more powerful capabilities. Accordingly, here is a brief guide to the relevant professional societies, the literature, and some widely available systems.

The Professional Societies

The Association for Computing Machinery (ACM) Special Interest Group on Symbolic and Algebraic Manipulation is the major iternational professional society for computer algebra. Their ACM SIGSAM Bulletin is the most concentrated and up-to-date source for abstracts and working papers together with announcements of meetings and systems. For information about joining, including special student rates, write the ACM at 1133 Avenue of the Americas, New York, NY 10036. Some European groups devoted to computer algebra are:

1. SAM-AFECT; Contact M. Bergman, Faculte des Sciences de Luminy, Case 901, 13009; or contact J. Calmet IMAG. B.P. 53, 38041 Grenoble Cedex, France.

2. NIGSAM; Contact Y. Sundblad, Department of Numerical Analysis and Computer Science, KTH S-10044 Stockholm, Sweden.

3. SEAS/SMC; Contact J S. van Hulzen,Twente University of Technology, P.O. Box 217, 7500 AE enschede, The Netherlands.

The Literature

Regrettably, there is not yet a textbook devoted to computer algebra, and the few textbooks that contain relevant material are rather advanced. Most of the information is sparsely scattered in research journals or in less accessible conference proceedings and reports. However, the following relatively accessible references contain surveys, bibliographies, and collections of articles that should serve as a good point of departure for exploring most facets of the literature.

1. ACM SIGSAM Bulletin, ACM, New York, all issues.

2. Communications of the ACM 14, No. 10, August 1971.

3. SIAM Journal on Computing 8, No. 3, August 1979.

4. Communications of the ACM 9, No. 10, August 1966.

5. Journal of the ACM 18, No. 4, October 1971.

6. P.S. Wang, editor, Proceedings of the 1981 ACM Symposium on Symbolic and, Algebraic Computation, ACM Order No. 505810, P.O. Box 64145, Baltimore, MD 21264, $23.

7. E.W. Ng, editor, Symbolic and Algebraic Computation, Lecture Notes in Computer Science, 72, Springer-Verlag, New York, 1979.

8. R.D. Jenks, editor, Proceedings of the 1976 ACM Symposium on Symbolic and Algebraic Computation, ACM, New York, 1976.

9. V.E. Lewis, editor, Proceedings of the 1979 MACSYMA User's Conference, M.I.T.

Laboratory for Computer Science, 545 Technology Square, Cambridge, Massachusetts, 1979.

10. C.M. Anderson, editor, Proceedings of the 1977 MACSYMA User's, Conference, NASA CP-2012, 1977.

11. Knuth, D.E., The Art of Computer Programming, Volume II, Seminumerical Algorithms, Addison- Wesley, Reading, Mass., 1980.

12. Yun, D.Y.Y. and Stoutemyer, "Symbolic Mathematical Computation," Encyclopedia of Computer Science and Technology, Supplementary Volume 15, J. Belzer, A.G. Holzman and A. Kent, editors, M. Dekker, New York, pp. 235-310.

Other Computer Algebra

ALGICALC has an analogous companion program that is about the same size and is distributed predominately for the same machines by the same software publishers: POLYCALC (APX-10127 and APX-20127) treats expressions that are equivalent to generalized polynomials in up to the 26 variables A through Z, any number of which can be unassigned. The exponents of variables can be fractional or negative numbers, but denominatiors with more than one term cannot be accommodated. Thus in comparison to ALGICALC, the class of allowable expressions is broader in some respects and narrower in others. Expansion, differentiation, integration, nonnumeric substitutions, and power-series truncation are supported, so the allowable operations are also broader in some respects and narrower in others. The net result is that together the two programs span a significantly larger range of problems than either package does alone.

Besides POLYCALC and ALGICALC, there are larger systems that provide more capabilities at the expense of greater demands on memory space, computer sophistication and cost. In contrast to ALGICALC and POLYCALC:

1. They require anywhere up to 200 times as much memory space.

2. They accommodate a significantly larger class of expressions, including general rational or algebraic expressions in any number of variables, perhaps with logarithmic, exponential and trigonometric extensions. Some of these systems can even accommodate equations, matrix or tensor expressions directly.

3. They provide their own indefinite-precision arithmetic to avoid the serious limitations of finite-precision, finite-magnitude arithmetic.

4. They accommodate much larger expressions having thousands of terms, with coefficients having hundreds of digits.

5. They provide a larger suite of built-in and optional transformations including, for example, multivariate factoring, partial-fraction, and various trigonometric expansions.

6. They provide convenient facilities permitting the user to write function definitions and simplification rules to extend the built-in capabilities by enlarging the allowable class of expressions or the variety of available transformations: In other words, they are programmble.

In approximate order of increasing memory requirements, here are some of the most widely available general-purpose systems that are currently supported:

1. muMATH-80(tm) is an interactive system that runs on microcomputers based on the 6502, 8080, 8085, Z80, or 8086 microprocessors, provided they have enough memory and an appropriate disk operating system. These include CP/M with at least 32 kilobytes of RAM memory, the Radio Shack TRS-DOS with at least 32 kilobytes of such memory, the Apple computer with at least 48 kilobytes of such memory or the IBM Personal Computer operating system with at least 64 kilobytes of such memory. muMATH is distributed to end users, computer stores and hardware manufacturers by Microsoft at 10800 N.E. Eighth, Suite 819, Bellevue, Washington 98004, and muMATH is also distributed by the authors: The Soft Warehouse, Box 11174, Honolulu, Hawaii 96828. The less common CP/M disk formats are available from Lifeboat Associates, 1651 Third Avenue, New York, N.Y. 10028.

2. SAC-2 is a non-interactive system that runs on any computer that can directly run a 1966 standard FORTRAN program of at least about 120 kilobytes. Information about SAC-2 is available from Professor George Collins, Computer Sciences Department, University of Wisconsin, 1210 West Dayton St., Madison, Wisconsin 53706.

3. FORMAC runs on any IBM 360 or 370 that can accommodate a PL/1 program of at least 150 kilobytes. FORMAC is semi-interactive on some operating systems. Information about FORMAC is available from Knut Bahr at GMD/IFV, D-6100, Darmstadt, Germany.

4. ALTRAN is a non-interactive system that runs on any computer that can directly run a 1966 standard FORTRAN program of at least about 270 kilobytes. Information about ALTRAN is available from the Computing Information Library, Bell Laboratories, 600 Mountain Avenue, Murray Hill, N.J. 07974.

5. REDUCE is an interactive system that runs on the IBM 360 or 370, DEC 10 or 20, Univac 1100 series, Control Data Cyber series, Burroughs 6700, and several other computers, requiring a minimum of about 350 kilobytes. For information about REDUCE, write Dr. Anthony Hearn, Rand Corporation, 1700 Main Street, Santa Monica, California 90401.

6. MACSYMA is an interactive system that runs on the DEC 10 or 20 as well as on the particularly sophisticated "LISP machine" personal computer. For information about the DEC version, contact Professor Joel Moses at M.I.T. For information about the LISP machine version, contact Symbolics Incorporated or The LISP Machine Incorporated in Santa Monica, California 90401.

7. SMP is an interactive system that runs on the DEC VAX or on other large-address-space machines supporting the UNIX operating system, including a compiler for the C language. (Sorry, PDP-11 machines do not have anywhere near enough address space.) For information, contact Stephen Cole at the California Institute of Technology in Pasadena, California.

Additional systems are announced in back issues of the ACM SIGSAM Bulletin.

COMPUTER ALGEBRA IN EDUCATION

It should be clear to anyone who has used a computer-algebra system that it has enormous potential for use in education as well as research. Not only can computer algebra make computing more attractive to mathematically inclined students; computer algebra can make mathematics more attractive to computer enthusiasts. Thus, computer algebra provides motivation between math and computer education.

Personal computers are becoming so prevalent that students, engineers, scientists, and mathematicians will soon be using computer algebra extensively. Moreover, it should not be long before general-purpose computer algebra is available on pocket calculators, because:

1. Several manufacturers now make low wattage "CMOS" versions of most popular 8-bit microprocessors, and CMOS versions of some 16-bit microprocessors are currently under development.

2. There are already hand-held terminals with 32 kilobytes of low wattage, read-write memory.

3. There are already hand-held calculators with sufficiently large low wattage liquid-crystals to display a reasonably large mathematical expression -- perhaps one term at a time.

Eventually some enterprising manufacturer will surely merge these three technologies, producing a hand-held calculator capable of running a full-fledged computer-algebra system such as muMATH. Thus, it behooves every math and computer science educator to explore how this revolutionary tool can be used to aid education.

It is undeniably true that most students are far more intrigued and motivated by the artificial intelligence and game playing applications of computers than by the accounting and numerical applications that currently account for most computer usage. Thus, it is advisable to exploit this strong preferential interest to help teach both mathematics and computer science. If more good math, science, and engineering students are attracted to computers and more good computer-oriented students are attracted to math, then more students will ultimately learn to use computers effectively for both numeric and nonnumeric purposes.

Computer algebra makes a highly motivating introductory computer programming course for math, science and engineering students. Computer algebra is also an ideal principal language for such students, because numbers and arithmetic comprise appreciably less than half of the kindergarden-through-calculus math curriculum. Moreover, the limited-precision integer and floating-point arithmetic typical of traditional programming languages is not the kind of arithmetic taught in this curriculum or used in everyday life.

Some educators may fear that computer algebra might cause algebraic skills to atrophy or prevent them from ever developing. Analogous concerns were undoubtedly expressed about Arabic numerals, multiplication tables, logarithms, Laplace transforms and numerical pocket calculators; but we have survived their convenience. The National Council for Teachers of Mathematics strongly supports the use of numerical pocket calculators in classrooms, and every reason for this support is even more true of computer algebra.

Automatic symbolic mathematics makes it possible for students to concentrate on basic mathematical concepts rather than spending an excessive amount of time mechanically performing transformations. Computer algebra lets students explore such fundamental concepts as commutativity, associativity, groups and rings. Moreover, the extensive algebraic capabilities of computer algebra enables students to investigate larger examples than is otherwise practical. Patterns thus revealed may suggest useful theorems. Conjectured patterns thus violated provide counterexamples against false hypotheses. Thus, computer algebra can contribute to teaching mathematical discovery.

Existing computer-algebra systems could also make other educational contributions:

1. Trace packages can be used to let students witness each step of an algebraic simplification, rather than merely the final result.

2. The very fact that algebra and calculus can be automated should encourage average and poor math students that the flashes of inspiration characteristic of quick students are unnecessary for those operations -- there is revealed hope for the slower, more methodical students.

3. For students who know how to program in the language in which the computer algebra packages are written, inspection of the underlying algorithms may help them learn methods for accomplishing the operations. Moreover, by programming extensions to the built-in facilities, students can reinforce understanding of the built-in and extended operations.

Many existing computer algebra systems can be used in the above ways right now. However, there is a potential for much more. In conjunction with a computer-aided instruction package, existing computer algebra systems could be used for extremely flexible and intelligent algebra drill, testing, and tutorial dialogue.

None of the existing computer algebra systems is by itself a computer-aided math instruction system. However, someone experienced in computer-aided instruction could design interactive math lessons or examinations to be used in conjunction with these existing computer algebra systems.

ERROR MESSAGES

The ALGICALC error messages are detailed and precise, indicating exactly where processing could not continue and why. For example, if the error was caused by an ill-formed expression, then ALGICALC states every category of character that is allowable at that point. This section lists all of these messages in alphabetical order, along with some of the most common causes.

Expected +, -, semicolon, digit, decimal point, variable or (:

Perhaps an operator or a parenthesis occurs where it is not allowed, such as in the entries "X**2", ")", "()", or "*". Alternatively, perhaps there is an unallowed character such as "&", "#" or "]".

Expected), operator, digit, decimal point, variable or (:

ALGICALC encountered the end of an expression while seeking a right parenthesis to match an earlier left parenthesis.

Expected nonzero divisor:

An attempt was made to divide by a factor that evaluates to zero, and division by zero is mathematically undefined. This message can also arise from an attempt to raise 0 to a negative power, which entails division by the corresponding positive, power of 0. Another common cause is substitution of a number that makes a denominator become 0.

Expected assignee other than X. See topic 3.

X cannot occur on the left side of an assignment. See sections on "Assignments" and "Substitutions".

Expected easier problem:

Either a number has overflowed or there is insufficient memory to determine the result.

Expected existing topic number:

The expression preceding a question mark must evaluate to one of the integer lesson numbers listed in the lesson menu.

Expected integer power because base is negative or has a negative leading coefficient:

ALGICALC cannot internally represent the imaginary numbers that correspond to fractional powers of negative numbers or coefficients.

Expected integer powered result:

ALGICALC cannot internally represent fractional powers of a nonnumeric expression.

Expected numeric power because base <> 1:

1 raised to any power is 1, but ALGICALC cannot otherwise treat exponential

subexpressions, such as 2^X or 2.71828^X.

Expected numeric term. See topic 6:

ALGICALC can only substitute numeric values for X, so the term to the right of "@" must evaluate to a number.

Expected return, semicolon, ampersand, operator, digit, decimal point, variable or (:

There are extra characters that cannot be incorporated in the indicated complete expression. Common causes are excess right parentheses or unallowed characters such as "$", "<", or "]".

Expected variable to be X or assigned. See topic 3.

A variable other than X occurs in an expression before being assigned a value. See section on "Assignments".

No convergence yet. Continue until START (Y/N)?

This question occurs when a factor has not been extracted within a few minutes, as explained in section "Factored Display".

Any other error messages are generated by ATARI BASIC in a manner that was unanticipated by The Soft Warehouse. See your ATARI BASIC Reference Manual.

QUICK REFERENCE SHEET

To enter problems --> enter data [RETURN]

To request HELP menu --> ? [RETURN]

To request a lesson --> lesson number ? [RETURN]

ASSIGNABLE VARIABLES --> A/a - W/w, Y/y - Z/z
UNASSIGNED VARIABLE --> X/x

ORDER OF OPERATIONS: ^ (exponentiation)
+, - (as signed numbers)
*, / (multiplication, division)
+, -, @, % (addition, subtraction, substitution, differentiation)

MULTIPLE EXPONENTIATION: right to left, when no parentheses
MULTIPLE MULTIPLICATION AND DIVISION: left to right, when no parentheses
MULTIPLE ADDITION, SUBTRACTION, SUBSTITUTION, DIFFERENTIATION: left to right, when no parentheses

Use PARENTHESES IN DIVISION to include more than one term (e.g., (x+7) as a denominator)

EXPONENTS --> integers only, for each subexpression

FACTORED DISPLAY --> end input with &

Formula to CONVERT IMAGINARY NUMBER display to corresponding quadratic factors: (x+a+bi)(x+a-bi) --> x^2 + 2ax + (a^2 + b^2)

Arithmetic limitations of ATARI BASIC
Magnitude limitation for integers to avoid roundoff errors:
LARGEST NONZERO MAGNITUDE: 10^98
SMALLEST NONZERO MAGNITUDE: 10^-98

ASSIGNMENTS: letter = expression

Letter can be A/a-W/w, Y/y-Z/z; X reserved as an unassigned variable.
Must assign a value to a variable (except X) before requesting ALGICALC to evaluate it in an expression

SUPPRESSED DISPLAY --> end input with ;

To enter MORE THAN ONE EXPRESSION/ASSIGNMENT PER LINE --> separate them with ;

SUBSTITUTION of numerical value for X: (sub)expression @ numeric value

DERIVATIVES of exponents or subexpressions with respect to X:
(sub)expression %
Repeat % to express higher order derivatives

APX ATARI® PROGRAM EXCHANGE
P.O. Box 3705
Santa Clara, CA 95055

Review Form

We're interested in your experiences with APX programs and documentation, both favorable and unfavorable. Many of our authors are eager to improve their programs if they know what you want. And, of course, we want to know about any bugs that slipped by us, so that the author can fix them. We also want to know whether our instructions are meeting your needs. You are our best source for suggesting improvements! Please help us by taking a moment to fill in this review sheet. Fold the sheet in thirds and seal it so that the address on the bottom of the back becomes the envelope front. Thank you for helping us!

1. Name and APX number of program.

2. If you have problems using the program, please describe them here.

3. What do you especially like about this program?

4. What do you think the program's weaknesses are?

5. How can the catalog description be more accurate or comprehensive?

6. On a scale of 1 to 10, 1 being "poor" and 10 being "excellent", please rate the following aspects of this program:

_____ Easy to use
_____ User-oriented (e.g., menus, prompts, clear language)
_____ Enjoyable
_____ Self-instructive
_____ Useful (non-game programs)
_____ Imaginative graphics and sound

7. Describe any technical errors you found in the user instructions (please give page numbers).

8. What did you especially like about the user instructions?

9. What revisions or additions would improve these instructions?

10. On a scale of 1 to 10, 1 representing "poor" and 10 representing "excellent", how would you rate the user instructions and why?

11. Other comments about the program or user instructions:

From

STAMP

ATARI Program Exchange
P.O. Box 3705
Santa Clara, CA 95055

[seal here]